Câu 2

```
dp = [[0] * (len(source_string) + 1) for i in range(len(target_string) + 1)]
for i in range(1, len(target_string) + 1):
    for j in range(1, len(source_string) + 1):
        if target_string[i - 1] == source_string[j - 1]:
            dp[i][j] = dp[i - 1][j - 1] + 1
        else:
            dp[i][j] = max(dp[i][j - 1], dp[i - 1][j])
```

Khởi tạo ma trận dp

Xét từng phần tử target_string với source_string. Áp dụng công thức:

$$LCSS(i,j) = \max \begin{cases} LCSS(i-1,j-1) & if\ x_i = y_i \\ LCSS(i-1,j)\ otherwise\ (no\ match\ on\ x_i) \\ LCSS(i,j-1)\ otherwise\ (no\ match\ on\ y_i) \end{cases}$$

```
i = len(target_string)
j = len(source_string)

operations_performed = []
while (i != 0 and j != 0):
    if  target_string[i - 1] == source_string[j - 1]:
        i -= 1
        j -= 1
    else:
        if dp[i][j - 1] < dp[i - 1][j]:
            operations_performed.append(('INSERT', target_string[i - 1]))
            i -= 1
        elif  dp[i][j - 1] < dp[i - 1][j - 1]:
            operations_performed.append(('INSERT', target_string[i - 1]))
            i -= 1
        elif  dp[i][j - 1] >= dp[i - 1][j]:
            operations_performed.append(('DELETE', source_string[j - 1]))
            j -= 1
        else:
            operations_performed.append(('SUBSTITUTE', target_string[i - 1], source_string[j - 1]))
            i -= 1
            j -= 1
while(j != 0):
    operations_performed.append(('DELETE', source_string[j - 1]))
    j -= 1

while(i != 0):
    operations_performed.append(('INSERT', target_string[i - 1]))
    i -= 1

operations_performed.reverse()
return [dp[len(target_string)][len(source_string)], operations_performed]
```

Dựa trên bài 1, tính các thông số INSERT, DELETE, SUBSTITUTE , thay đổi điều kiện if cho pguf hợp với công thức LCS. Nếu 2 giá trị bằng nhau thì thêm SUBSTITUDE vào danh sách, đồng thời giảm i,j. Tương tự cho 2 trường hợp kia với 2 giá trị không bằng nhau.

```
if __name__ == "__main__":

    s1 = "ACADB"
    s2 = "CBDA"
    distance, operations_performed = lcss_dp(s1, s2)

    insertions, deletions, substitutions = 0, 0, 0
    for i in operations_performed:
        if i[0] == 'INSERT':
            insertions += 1
        elif i[0] == 'DELETE':
            deletions += 1
        else:
            substitutions += 1

  # Print the results
    print("LCSS distance : {}".format(distance))
    print("Number of insertions : {}".format(insertions))
    print("Number of deletions : {}".format(deletions))
    print("Number of substitutions : {}".format(substitutions))
    print("Total number of operations : {}".format(insertions + deletions + substitutions))

    print("Actual operations :")
    for i in range(len(operations_performed)):
        if operations_performed[i][0] == 'INSERT':
            print("{}) {} : {}".format(i + 1, operations_performed[i][0], operations_performed[i][1]))
        elif operations_performed[i][0] == 'DELETE':
            print("{}) {} : {}".format(i + 1, operations_performed[i][0], operations_performed[i][1]))
        else:

          print("{}) {} : {} by {}".format(i + 1, operations_performed[i][0], operations_performed[i][1], operations_performed[i]
```

Tương tự như câu 1 nhưng gán các giá trị đầu vào cố định.

Câu 3

```
n = len(ts_a)
m = len(ts_b)

# Tạo ma trận
distance = np.zeros((n, m))
path = np.zeros((n, m), dtype=np.int)
distance[0, 0] = abs(ts_a[0] - ts_b[0])


for i in range(1, n):
    distance[i, 0] = distance[i - 1, 0] + abs(ts_a[i] - ts_b[0])
    path[i, 0] = 1  # 'INSERT'
for j in range(1, m):
    distance[0, j] = distance[0, j - 1] + abs(ts_a[0] - ts_b[j])
    path[0, j] = 2  #  'DELETE'

for i in range(1, n):
    for j in range(1, m):
        cost = abs(ts_a[i] - ts_b[j])
        distance[i, j] = cost + min(distance[i - 1, j], distance[i, j - 1], distance[i - 1, j - 1])
        if min(distance[i - 1, j], distance[i, j - 1], distance[i - 1, j - 1]) == distance[i - 1, j - 1]:
            path[i, j] = 3  # 'SUBSTITUTE'
        elif min(distance[i - 1, j], distance[i, j - 1], distance[i - 1, j - 1]) == distance[i - 1, j]:
            path[i, j] = 1  #  'INSERT'
        else:
            path[i, j] = 2  #  'DELETE'
```

Tạo ma trận path với số chiều, cột là độ dài 2 danh sách đầu vào

Distance là khoảng cách thời gian được tính theo công thức(trị tuyệt đối từng giá trị đầu vào của 2 danh sách $M(\bar{X}_i, \bar{Y}_i) = |x_i - y_i| + M(\overline{X_{i-1}}, \overline{Y_{i-1}})$

$$DTW(i,j) = distance(A_i, B_j) + \min(DTW(i,j-1), DTW(i-1,j), DTW(i-1,j-1))$$

$$\text{với } distance(A_i, B_j) = |A_i - B_j|$$

Xét từng dòng với cột 0 và ngược lại, cột với dòng 0 thông qua 2 vòng lặp for. Nhằm mục đích khởi tạo giá trị đầu để tích các giá trị tiếp theo.

$$DTW(i,j) = distance(x_i, y_j) + min\begin{cases} DTW(i,j-1) & repeat\ x_i \\ DTW(i-1,j) & repeat\ y_j \\ DTW(i-1,j-1) & repeat\ neither \end{cases}$$

Trong khi xét từng giá trị của i với j. Nếu các giá trị min bằng distance[i - 1, j - 1] thì path tại vị trí đó đại diện cho SUBSTITUTE, tương tự cho DELETE và INSERT.

nếu min = distance[i - 1, j] hoặc min = distance[i , j-1].

```
    i = n - 1
    j = m - 1
    operations_performed = []
    while (i != 0 and j != 0):
        if path[i, j] == 3:  # SUBSTITUTE
            operations_performed.append(('SUBSTITUTE', ts_b[j], ts_a[i]))
            i -= 1
            j -= 1
        elif path[i, j] == 1:  # INSERT
            operations_performed.append(('INSERT', ts_a[i]))
            i -= 1
        else:  # DELETE
            operations_performed.append(('DELETE', ts_b[j]))
            j -= 1
    operations_performed.reverse()

    return distance[n - 1, m - 1], operations_performed
```

Tương tự bài 1,2 nhưng xét path với các giá trị đã gán ở trên để tính 3 thông số.

```python
if __name__ == "__main__":
    ts_a = [1,7,4,8,2,9,6,5,2,0]
    ts_b = [1,2,8,5,5,1,9,4,6,5]

  # Find the minimum edit distance and the operation performed
    distance, operations_performed = dynamic_time_warping(ts_a, ts_b)

  # Count the number of individual operations
    insertions, deletions, substitutions = 0, 0, 0
    for i in operations_performed:
        if i[0] == 'INSERT':
            insertions += 1
        elif i[0] == 'DELETE':
            deletions += 1
        else:
            substitutions += 1

  # Print the results
    print("Minimum edit distance : {}".format(distance))
    print("Number of insertions : {}".format(insertions))
    print("Number of deletions : {}".format(deletions))
    print("Number of substitutions : {}".format(substitutions))
    print("Total number of operations : {}".format(insertions + deletions + substitutions))

    print("Actual operations :")
    for i in range(len(operations_performed)):
        if operations_performed[i][0] == 'INSERT':
            print("{}) {} : {}".format(i + 1, operations_performed[i][0], operations_performed[i][1]))
        elif operations_performed[i][0] == 'DELETE':
            print("{}) {} : {}".format(i + 1, operations_performed[i][0], operations_performed[i][1]))
        else:

            print("{}) {} : {} by {}".format(i + 1, operations_performed[i][0], operations_performed[i][1], operations_performed[
```

Tương tự bài 1,2, gán giá trị cố định là ts_a và ts_b